A Logic of Trust for Reasoning about Delegation and Revocation

Marcos Cramer
University of Luxembourg

Diego Agustín Ambrossio
University of Luxembourg

Pieter Van Hertum
KU Leuven

ABSTRACT

In ownership-based access control frameworks with the possibility of delegating permissions and administrative rights, chains of delegated accesses will form. There are different ways to treat these delegation chains when revoking rights, which give rise to different revocation schemes. Hagström et al. [8] proposed a framework for classifying revocation schemes, in which the different revocation schemes are defined graph-theoretically; they motivate the revocation schemes in this framework by presenting various scenarios in which the agents have different reasons for revocating. This paper is based on the observation that there are some problems with Hagström et al.'s definitions of the revocation schemes, which have led us to propose a refined framework with new graph-theoretic definitions of the revocation schemes. In order to formally study the merits and demerits of various definitions of revocation schemes, we propose to apply the axiomatic method originating in social choice theory to revocation schemes. For formulating an axiom, i.e. a desirable property of revocation frameworks, we propose a logic, *Trust Delegation Logic* (*TDL*), with which one can formalize the different reasons an agent may have for performing a revocation. We show that our refined graph-theoretic definitions of the revocation schemes, unlike Hagström et al.'s original definitions, satisfy the desirable property that can be formulated using TDL.

Categories and Subject Descriptors

K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

Keywords

delegation, revocation, trust, logic, access control

1. INTRODUCTION

In ownership-based frameworks for access control, it is common to allow principals (users or processes) to grant both permissions and administrative rights to other principals in the system. Often it is desirable to grant a principal the right to further grant permissions and administrative rights to other principals. This may lead to delegation chains starting at a *source of authority* (the owner of a resource) and passing on certain permissions to other principals in the chain.

Furthermore, such frameworks commonly allow a principal to revoke a permission that she granted to another principal. Depending on the reasons for the revocation, different ways to treat the chain of principals whose permissions depended on the second principal's delegation rights can be desirable. For example, if one is revoking a permission given to an employee because he is moving to another position in the company, it makes sense to keep in place the permissions of principals who received their permissions from this employee; but if one is revoking a permission from a user who has abused his rights and is hence distrusted by the user who granted the permission, it makes sense to delete the permissions of principals who received their permission from this user. Any algorithm that determines which permissions to keep intact and which permissions to delete when revoking a permission is called a *revocation scheme*. Revocation schemes are usually defined in a graph-theoretical way on the graph that represents which authorizations between the principals are intact.

Hagström et al. [8] have presented a framework for classifying possible revocation schemes along three different dimensions: the extent of the revocation to other grantees (propagation), the effect on other grants to the same grantee (dominance), and the permanence of the negation of rights (resilience). Since there are two options along each dimension, there are in total eight different revocation schemes in Hagström et al.'s framework. This classification was based on revocation schemes that had been implemented in database management systems [7, 6, 2, 3]. The framework's design decisions are carried over from these database management systems and are often not fully motivated. Furthermore, the behaviour of the revocation schemes is dependent on the conflict resolution policy of the system, which is not integrated into the framework.

We identify a number of problems with Hagström et al.'s framework and the definitions of the revocation schemes included in the framework. This motivates our refined framework, in which the conflict resolution policy is integrated into the framework, and in which the graph-theoretic definitions of the revocation schemes have been modified.

In order to avoid that our refined framework turns out to have undesirable properties like those we identified in Hagström et al.'s framework, we propose to formally study the merits and demerits of various definitions of revocation schemes using the axiomatic method originating in social choice theory. Which behaviour is desirable for a revocation scheme depends on the reasons for performing the revocation. So in order to formulate an axiom, i.e. a desirable property of revocation schemes, we propose a logic, *Trust Delegation Logic* (*TDL*), with which one can formalize the differ-

SACMAT'15, June1–3, 2015, Vienna, Austria.

ACM 978-1-4503-3556-0/15/06 ...$15.00.
DOI: http://dx.doi.org/10.1145/2752952.2752968.

ent reasons an agent may have for performing a revocation. We show that our modified graph-theoretic definitions of the revocation schemes, unlike Hagström et al.'s original definitions, satisfy the desirable property that can be formulated using TDL.

The rest of the paper is structured as follows. In Section 2, we discuss related work, giving an overview of Hagström et al.'s framework as well as of the conflict resolution policies proposed in the literature. In Section 3 we motivate and define a refinement to Hagström et al.'s framework. In Section 4, we consider the diverse reasons for revocating on some example scenarios, and sketch how these reasons can be used to formulate the desirable behaviour of the revocation schemes. In Section 5, we motivate and define Trust Delegation Logic (TDL). In Section 6 we illustrate how the scenarios discussed in Section 4 can be formalized in TDL. In Section 7 we use TDL to formally formulate a desirable property for revocation frameworks, which our revocation framework satisfies. After discussing possible further work on the topic of this paper in Section 8, we conclude the paper in Section 9.

2. RELATED WORK

Hagström et al. [8] have introduced three dimensions according to which revocation schemes can be classified. These are called *propagation*, *dominance* and *resilience*:

Propagation. The decision of a principal i to revoke an authorization previously granted to a principal j may either be intended to affect only the direct recipient j or to propagate and affect all the other users in turn authorized by j. In the first case, we say that the revocation is *local*, in the second case that it is *global*.

Dominance. This dimension deals with the case when a principal losing a permission in a revocation still has permissions from other grantors. If these other grantors' revocation rights are dependent on the revoker, the revoker can dominate over these grantors and revoke the permissions from them. This is called a *strong* revocation. The revoker can also choose to make a *weak* revocation, where permissions from other grantors to a principal losing a permission are kept.

Resilience. This dimension distinguishes revocation by removal (deletion) of positive authorizations from revocation by issuing a negative authorization which just inactivates positive authorizations. In the first case another principal may grant a similar authorization to the one that had been revoked, so the effect of the revocation does not persist in time. In the second case a negative authorization will overrule any (new) positive permission given to the same principal, so its effect will remain until the negative permission is revoked. We call a revocation of the first kind a *delete* or *non-resilient* revocation, and a revocation of the second kind a *negative* or *resilient* revocation.

Since there are two possible choices along each dimension, Hagström et al.'s framework allows for eight different revocation schemes.

Delegation frameworks that allow issuing negative authorization can bring about a state in which a conflict may arise. If a principal is granted both a positive and a negative authorization for the same object, then we say that these two authorizations *conflict* each other. A system's *conflict resolution policy* determines how to resolve such a conflict. Here is a list of possible conflict resolution policies as described by Ruan and Varadharajan [11]:

Negative-takes-precedence: If there is a conflict occurring on the authorization for some object, the negative authorizations will take precedence over the positive one.

Positive-takes-precedence: Positive authorizations from i to j take precedence over negative authorizations from k to j for all $k \neq i$. This means that a negative authorization from i to j directly inactivates only positive authorizations from i to j, and leaves other permission to j active.

Strong-and-Weak: Authorizations are categorized in two types, strong and weak. The strong authorizations always take precedence over the weak ones. Conflicts among strong authorizations are not allowed. In conflicts between weak authorizations negative ones take precedence. Note that the intended meaning of *strong* and *weak* in this policy differs from their meaning in Hagström et al.'s dominance dimension.

Time-takes-precedence: New authorizations take precedence over previously existing ones. Note that this policy will make negative authorizations non-resilient.

Predecessor-takes-precedence: If the principal i delegates (possibly transitively) some right to principal j, then authorizations issued by i to some other principal k concerning that right will take precedence over authorizations issued by j to k. In other words, the priority of subjects decreases as the privilege is delegated forward.

Hagström et al. assume the system to have either a negative-takes-precedence or a positive-takes-precedence conflict resolution policy. Note that under a negative-takes-precedence policy, a negative revocations on principal k dominates all positive authorizations to k, so that the difference between weak and stron negative revocations disappears.

3. REFINING THE REVOCATION FRAMEWORK

In this section we first analyze some problems with the revocation framework by Hagström et al. [8]. While analyzing the problems, we already informally sketch how we propose to solve them. Next we define a refined revocation framework in which all of these problems have been solved.

3.1 Problems with Hagström et al.'s framework

(1) In Hagström et al.'s framework, strong global revocations will propagate forward dominating over all the existing delegation chains, making them even stronger than desired. We illustrated this by an example:

EXAMPLE 1. *User A issues an authorization to users B and C. B also grants this authorization to C. If a strong global delete revocation (in Hagström et al.'s sense) is performed over the authorization from A to B, then the authorization A granted to C is also deleted. But since A granted this authorization to C independently from B, it seems unjustified to delete it (Hagström et al. give no motivation for this behaviour).*

(2) In Hagström et al.'s framework, the choice of a conflict resolution policy is not incorporated into the revocation framwork, even though it affects the behaviour of the dominance dimension. We extend the dominance dimension to incorporate the choice of how to resolve conflicts between positive and negative authorizations in the revocation framework. In our refined framework, there are three choices along the dominance dimension:

- **weak:** The principal performing the revocation only dominates over direct authorizations granted by herself, authorizations from other grantors are kept intact.
- **predecessor-takes-precedence (p-t-p):** The principal performing the revocation dominates over other grantors' authorizations that are dependent on her.
- **strong:** The principal performing the revocation dominates over all other grantors' authorizations.

Note that we now use the terminology from conflict-resolution policies as presented in [11] and in Section 2 for the choices on the

dominance dimension. Hence "strong" now has a different meaning than in Hagström et al.'s framework: As long as Hagström et al.'s framework is combined with a positive-takes-precedence policy, the strong revocations in their framework have the same force as our p-t-p revocations. The strength of our strong revocations can only be achieved in Hagström et al.'s framework by combining it with a negative-takes-precedence policy.

It is not desirable to allow all users who have a delegation right to perform strong revocations. Hence we include in our framework the possibility for a principal to grant to another principal a special right to perform strong revocations to other users.

(3) In Hagström et al.'s framework, delete revocations are supposed be non-resilient, which according to Hagström et al. means that "another user may issue the same permission that was just revoked, and the effect of the revocation disappears". This property fails to be satisfied in global deletion revocations, as illustrated by this example:

EXAMPLE 2. *User A issues an authorization to user B, and B further grant this authorization to C. If A deletes the authorization given to B, then the authorization from B to C is also deleted. Reissuing the authorization from A to B will not re-instate the authorization from B to C as before the revocation.*

To avoid this problem in our framework, when a delete is performed, we do not delete the forward chain, but just inactivate it.

(4) Hagström et al. motivate the distinction between delete and negative revocations mainly through the notion of resilience as defined in Section 2. However, in weak revocations there can be no difference between a resilient and a non-resilient revocation, since a weak revocation does not affect authorizations issued by others than the revoker. They motivate the usage of weak negatives by pointing out that they are useful for temporary revocations. But since in our framework the forward chain does not get deleted in a delete revocation (see point (3)), a delete can also be easily undone, so that a delete revocation is a sensible choice even when the revocation is likely to be only temporary. Hence we do not need weak negative revocations.

Furthermore, p-t-p and strong deletes would have undesirable effects, as illustrated by the following example:

EXAMPLE 3. *User A issues an authorization to user B, and gives user C the right to perform strong revocations. User C performs a strong delete on B, removing without traces the authorization provided to B by A. Later A realizes that C cannot be trusted to perform strong revocations, and takes away B's right to do so. Even though C can no longer perform strong revocations, the effect of his strong delete persist: B does not have the right originally issued to him by A until someone issues a new authorization to him.*

Hence we do not have a p-t-p or strong delete revocation in our framework, but instead have the distinction between a resilient and a non-resilient negative for p-t-p and strong revocations.

To conclude, if the dominance of a revocation is p-t-p or strong, there are two options along the resilience dimension, non-resilient and resilient. But if the dominance is weak, there is no choice along the resilience dimension, and the revocation is characterized as a "weak delete". So there are five possible choices to be made along the dominance and resilience dimensions: weak delete, p-t-p non-resilient, p-t-p resilient, strong non-resilient, and strong resilient.

(5) Hagström et al. do not allow negative authorizations to be inactivated. The reason they give is that they "do not want a revocation to result in a subject having more permissions than before the revocation". However, the deletion of negative authorizations is allowed, even though it may have the same effect. We do allow negative authorizations to be inactivated, but the only kind of revocation that can result in a subject having more permissions than before is a revocation of someone's right to perform strong revocations, and in this case this is a desirable property.

3.2 The refined framework

Let **S** be the set of principals (subjects) in the system, let **O** be the set of objects in the system and let **A** be the set of access types. For every object $o \in \mathbf{O}$, there is a *source of authority* (SOA), i.e. the manager of object o.

For any $\alpha \in \mathbf{A}$ and $o \in \mathbf{O}$, the SOA of o can grant the right to access α on object o to other principals in the system. Secondly, the SOA can delegate this granting right further. Thirdly, the SOA can grant the right to perform strong revocations and to delegate this right further. Accordingly we have three *permissions*: *access right* (A), *delegation right* (D) and *strong revocation right* (S). We assume that delegation right implies access right. The set $\{A, D, S\}$ of permissions is denoted by **P**.

Additionally to positive authorizations ($+$), our framework admits four different types of negative authorizations, *p-t-p resilient negative* ($-_{\text{PR}}$), *p-t-p non-resilient negative* ($-_{\text{PN}}$), *strong resilient negative* ($-_{\text{SR}}$) and *strong non-resilient negative* ($-_{\text{SN}}$). The set $\{+, -_{\text{PR}}, -_{\text{PN}}, -_{\text{SR}}, -_{\text{SN}}\}$ of authorization types is denoted by **T**.

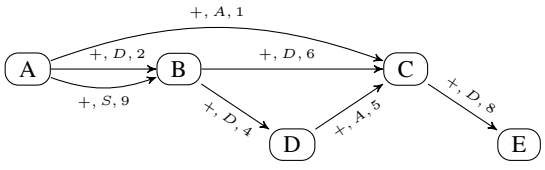
DEFINITION 1. *An* authorization *is a tuple* $(i, j, \alpha, o, \tau, \pi, t)$*, where* $i, j \in \mathbf{S}$*,* $\alpha \in \mathbf{A}$*,* $o \in \mathbf{O}$*,* $\tau \in \mathbf{T}$*,* $\pi \in \mathbf{P}$*,* $t \in \mathbb{Z}$*.*

The meaning of an authorization $(i, j, \alpha, o, \tau, \pi, t)$ is that at time point t principal i has granted to principal j an authorization of type τ for permission π concerning access type α on object o. We assume that all authorizations in the system are stored in an *authorization specification*. There is no interaction between the rights of principals concerning different access-object pairs (α, o). For this reason, we can consider α and o to be fixed for the rest of the paper, and can simplify $(i, j, \alpha, o, \tau, \pi, t)$ to (i, j, τ, π, t).

Since delegation right implies access right, an authorization $(i, j, +, D, t)$ can only be issued if an authorization $(i, j, +, A, t)$ is also issued. By taking the contrapositive, the connection is reversed for negative authorizations: For $\tau \neq +$, an authorization (i, j, τ, A, t) can only be issued if an authorization (i, j, τ, D, t) is issued.

We visualize an authorization specification by a labelled directed graph, as in Example 4, in which A is the SOA. For every authorization (i, j, τ, π, t) in the authorization specification, this graph contains an edge from i to j labelled τ, π, t. We refrain from showing the authorizations that can be implied to exist by the rules specified in the previous paragraph.

EXAMPLE 4. *An authorization specification*



A negative authorization can inactivate other authorizations in the authorization specification. Which authorizations get inactivated by a negative authorization depends on which type of negative authorization it is. There are three basic ideas governing the inactivation of authorizations: Firstly, non-resilient authorizations can only inactivate previously issued authorizations, whereas resilient authorizations can also inactivate authorization issued after the negative authorization. Secondly, a strong negative authorization from i to j inactivates every positive authorization from some

principal k to j, whereas a p-t-p authorization from i to j only inactivates an authorizations from k to j if k is dependent on i. Thirdly, any authorization that is no longer connected back to the SOA through active authorizations is inactivated.

In order to formally specify which authorizations get inactivated when issuing a negative authorization, we simultaneuosly define the notions of an authorization being *active* and an authorization being *directly inactivated* in Definitions 2 and 3.[1] The auxiliary notion of a directly inactivated authorization captures the idea of an authorization from k to j being inactivated by a strong negative authorization from i to j.

DEFINITION 2. *An authorization (i, j, τ, π, t) is* active *if it is not* directly inactivated *and there are principals $p_1, \ldots, p_n, p_{n+1}$ and integers $t_1, \ldots, t_n$ satisfying the following properties:*

(i) $p_1 = SOA$, $p_n = i$, $p_{n+1} = j$ and $t_n = t$;

(ii) for $1 \leq l < n$ there is an authorization $(p_l, p_{l+1}, +, \pi', t_l)$ that is not directly inactivated*, where $\pi' = S$ if either $\tau \in \{-_{\text{SR}}, -_{\text{SN}}\}$ or $\pi = S$, and $\pi' = D$ otherwise;*

(iii) there do not exist l, m with $1 \leq l \leq m \leq n$ and an authorization $(p_l, p_{m+1}, \tau', \pi'', t')$ such that $\pi'' = \pi$ and $\tau = +$ if $m = n$, and $\pi'' = \pi'$ otherwise, and such that either $\tau' = -_{\text{PN}}$ and $t' > t_m$ or $\tau' = -_{\text{PR}}$.

DEFINITION 3. *An authorization $(i, j, +, \pi, t)$ is* directly inactivated *if there is an* active *authorization $(k_1, j, -_{\text{SR}}, \pi, t_1)$ or there is an* active *authorization $(k_2, j, -_{\text{SN}}, \pi, t_2)$ with $t_2 > t$.*

A principal j has the right to access of type α on object o iff j is the SOA or there is an active authorization of the form $(i, j, \alpha, o, +, A, t)$ or $(i, j, \alpha, o, +, D, t)$. j has the right to perform strong revocations concerning action α on object o iff j is the SOA or there is an active authorization of the form $(i, j, \alpha, o, +, S, t)$. Strong negative authorizations towards the SOA are disallowed.

In Definition 4, we define the ten revocation schemes of our refined framework. We use W, P, S, L, G, N, R and D as abbreviations for *weak*, *p-t-p (predecessor-takes-precedence)*, *strong*, *local*, *global*, *non-resilient*, *resilient* and *delete* respectively. Note that when defining the revocation schemes, we do not need to specify which of the authorizations get inactivated, because Definition 2 already tells us what to inactivate. Hence we just specify which authorizations get added and/or deleted.

DEFINITION 4. *Let i, j be principals, and $\pi \in \{A, D, S\}$.*

- *A* WGD revocation for permission π from i to j at time t *consists of deleting any authorization of the form $(i, j, +, \pi, t')$.*
- *For $\delta \in \{\text{P,S}\}$ and $\rho \in \{\text{N,R}\}$, a $\delta\text{G}\rho$* revocation for permission π from i to j at time t *consists of issuing the negative authorization $(i, j, -_{\delta\rho}, \pi, t)$.*
- *For $(\delta, \rho) \in \{(\text{W,D}), (\text{P,N}), (\text{P,R}), (\text{S,N}), (\text{S,R})\}$, a $\delta\text{L}\rho$* revocation for permission π from i to j at time t *consists of deleting and adding the same revocations as in a $\delta\text{G}\rho$ revocation from i to j, and – if π is D or S – additionally adding an authorization (i, l, τ, π', t') for every authorization (j, l, τ, π', t') such that $\pi' = S$ if $\pi = S$, and π' is either D or A if $\pi = D$.*

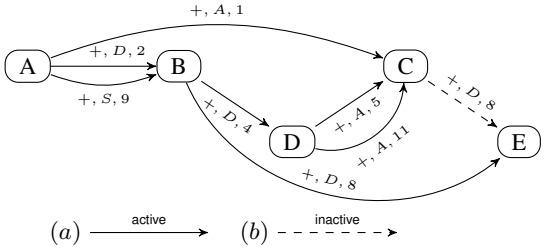
Since delegation right implies access right, a revocation for permission A can only be performed if the corresponding revocation for permission D is performed at the same time.

[1]These definitions inductively depend on each other. They should be read as an inductive definition with the well-founded semantics [5]. In Appendix A we discuss some of the issues resulting from the inductive interdependence of these definitions.

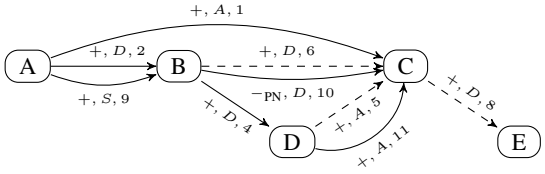
3.3 Examples of revocations

Here are three examples for different revocations from B to C on the authorization specification from Example 4. Examples for the other seven revocation schemes of our framework can be found in Appendix B. In all examples, we show the effect of simultaneous revocations for permissions A and D. In order to illustrate better the difference between resilient and non-resilient revocations, we show the state of the authorization specification after C reissues the previously issued authorization to D after the revocation.

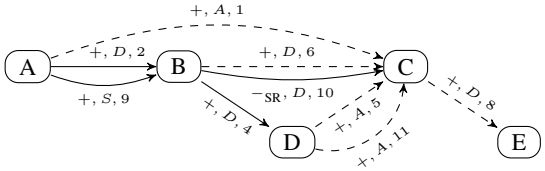
EXAMPLE 5. *Weak Local Delete revocation from* B *to* C



EXAMPLE 6. *P-t-p Global Non-resilient revocation from* B *to* C



EXAMPLE 7. *Strong Global Resilient revocation from* B *to* C



4. REASONS FOR REVOCATING

Hagström et al. have motivated the variety of revocation schemes by sketching various scenarios in which the principals performing the revocation have different reasons for revocating, so that different behaviour of the revocation is desirable. In order to study the desirable behaviour of the various revocation schemes, in this section we first present some scenarios, which we use to illustrate the different reasons the revocator has to perform the revocation, based on her level of trust or distrust towards the revokee. Having presented the scenarios, we informally sketch how we want to define the desirable behaviour of the various revocation schemes. These ideas will be formalized in subsequent sections.

4.1 Four scenarios

SCENARIO 1. *User A caught user C leaking information to a third-party. A revokes C's rights, ensuring that C cannot be given access by other users in the system.*

In this scenario user A had trusted user C in the past, thus issuing him an authorization, but now A distrusts principal C due the fact that he has leaked information to a third-party. So A will perform a *P-t-p Global Resilient* revocation, and – if she has strong revocation

right – additionally a *Strong Global Resilient* revocation, both in order to remove the authorization she had granted and to forbid as many other principals as possible to grant new authorizations to C.

SCENARIO 2. *User C is leaving to join the rival company. When user A notices the situation, she preemptively blocks C's capabilities (but keeping the authorizations previously issued by C).*

In this scenario user A had trusted user C in the past, thus issuing him an authorization. Since C is leaving to the rival company, A now distrusts C to access files or to newly delegate access right to others, but since A never misused any rights, A still has trust in the delegation authorization previously issued by C. So A will perform a *P-t-p Local Resilient* revocation and – if possible – a *Strong Local Resilient* revocation, in order to remove the authorizations that had been granted to C and to forbid as many other principals as possible to grant new authorizations to C, at the same time preserving the effect of authorizations that C had previously delegated.

SCENARIO 3. *User A hears the rumour that user B has received a bribe, but A does not know whether the rumour is true. Upon informing other users, A revokes B's rights, allowing other users to re-issue them.*

In this scenario, A no longer trusts B since she has heard the rumour about B, so she will revoke B's rights. But since A does not know whether the rumour is true, A allows other users to give the rights back to B (A will tell others about the rumour and trusts them to only give the rights back to B if they know the rumour to be false). So A will perform a *P-t-p Global Non-resilient* revocation and – if possible – a *Strong Global Non-resilient* revocation, in order to inactivate the previously issued authorizations granted to B, at the same time allowing other users to newly grant authorizations to B.

SCENARIO 4. *User A is revising the authorizations she had granted in the past. During the process A finds an authorization to user C, whom A does not remember.*

In this scenario user A had trusted user C in the past, thus issuing him an authorization, but now A neither trusts nor distrusts C, as she has no recollection of who C is or why the authorization had been granted. So A will perform a *Weak Global Delete* in order to remove the authorization she had granted to C without affecting authorizations granted to C by other users.

4.2 Desirable behaviour of revocation schemes

When explaining the above scenarios, we referred to the level and manner of trust or distrust between the revoker and the revokee in order to motivate the choice of different revocation schemes in different scenarios. The main novel idea of this paper is to formalize this reasoning about trust, delegation and revocation in such a way that we can formulate desirable properties that graph-theoretic definitions of revocation schemes should satisfy. Before we present this formalization, we will – for the rest of this section – first sketch the ideas behind this formalization and these desirable properties.

In Section 5, we will define *Trust Delegation Logic* (*TDL*), a logic that allows us to reason about the different levels and manners of trust or distrust that we find in the above four scenarios. One central idea in this logic is that A grants B the right to further delegate some right only if A trusts B to make correct judgments about who should be given that right. By expressing her trust in B to make correct judgments about something, A commits herselves to the truth of judgments that she has not made herself, namely the judgments that B has committed himself to. When A makes a judgment herself, we say that A has *explicit belief* in the judgment, whereas a judgment that A is committed to in the light of a principal trusted by A believing the statement is an *implicit belief* of A. Trust of principal A in principal B is modelled as A's belief in B's trustworthy. Depending on whether A's belief is explicit or implicit, we can also call this trust explicit or implicit. For example, if A expresses trust in B concerning the action of expressing trust in other principals, and B expresses trust in C, then A explicitly trusts B and implicitly trusts C.

A further central idea is that a principal A should have access right of access type α iff the SOA of that object trusts A, either explicitly or implicitly, concerning access α. Delegation chains correspond to chains of principals along whom an implicit trust in some principal can project upwards towards the SOA. A revocation takes place when at some point along such a chain of principals, a principal stops trusting in the next principal on the chain, thus disabling this upward projection of implicit trust.

TDL allows us to model different ways in which a principal can stop trusting or start distrusting another principal. Some of these ways have been illustrated in the above four scenarios. The various revocation schemes correspond to these various ways of stopping to trust.

Given these explanations, we can now sketch how TDL allows us to formulate a desirable property for graph-theoretic definitions of revocation schemes: The graph-theoretic definitions of the revocation schemes should be such that for any given delegation and revocation interaction between the principals, an active authorization to a principal A should exist in a graph if and only if – translating the delegation and revocation behaviour to TDL – the SOA believes A to be trustworthy for the access in question.

5. A LOGIC FOR REASONING ABOUT DELEGATION AND REVOCATION

In this section we present a logic for formalizing the reasons for revoking delegations. This logic, which we call *Trust Delegation Logic* (*TDL*), is a first-order multi-modal logic with both classical negation and negation-as-failure. TDL formalizes both the relation of trust between principals and the action of announcing one's trust in another principal by delegating some right to him/her/it. In developing TDL, we have taken over some ideas from [4] and [1].

We first define the syntax of TDL. Next we motivate its constructs and some of its axioms by sketching how we apply TDL for modelling delegation and revocation. After that we formally define the proof theory of TDL, briefly motivating the remaining axioms. TDL is only defined proof-theoretically, i.e. it does not have a formal semantics, since this is not needed for the application that we have in mind.

5.1 TDL syntax

There are five types of objects, *principal*, *access*, *time point*, *set of principal-time-pairs* and *announcement modality*. SOA is a constant symbol of type *principal*, $\emptyset$ is a constant symbol of type *set of principal-time-pairs*, B, $\not{B}$, K and $\not{K}$ are constants of type *announcement modality*, and ∞ and all integers are constants of type *time point*. *scons* is a ternary function symbol taking a term of type *principal*, a term of type *time* and a term of type *set of principal-time-pairs*, and returning a term of type *set of principal-time-pairs*. We use t, t', t_2, t_3 as *time point* variables, i, j, k as *principal* variables, $\Sigma, \Sigma', \Sigma_2$ as variables of type *set of principal-time-pairs*, α as an *action* variable, m as an *announcement modality* variable, and x as a variable of arbitrary type. Formulae of TDL are defined by

the following EBNF rule:

$$\varphi ::= \neg\varphi \mid {\sim}\varphi \mid (\varphi \wedge \varphi) \mid \forall x\ \varphi \mid T_i\alpha \mid T_i^t\varphi \mid T_i\mathbb{D}\alpha \mid T_i\mathbb{S}\alpha \mid B^t_{i,\Sigma}\varphi \mid K^t_{i,\Sigma}\varphi \mid I^t_i m\varphi \mid R^t_{i,\Sigma}m\varphi \mid r^t_i a \mid t > t' \mid (i,t) \in \Sigma$$

We write $\varphi \rightarrow \psi$ for $\neg(\varphi \wedge \neg\psi)$, $\varphi \leftrightarrow \psi$ for $(\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$, $\exists t > t'\ \varphi$ for $\exists t\ (t > t' \wedge \varphi)$, and $\exists x\ \varphi$ for $\neg\forall x\ \neg\varphi$. We drop brackets according to usual conventions. If φ is of the form $\neg\psi$, $\bar{\varphi}$ denotes ψ. Else $\bar{\varphi}$ denotes $\neg\varphi$.

While $\neg\varphi$ is the classical negation of φ, ${\sim}\varphi$ is negation as failure, i.e. ${\sim}\varphi$ is provable when φ is not provable.

$\emptyset$ refers to the empty set, and given a set Σ of principal-time-pairs, *scons*(i,t,Σ) refers to the set $\Sigma \cup \{(i,t)\}$ (*scons* stands for *set constructor* [9]). The following two axioms model this behaviour of $\emptyset$ and *scons*:

$$(\emptyset)\ \forall i\ \forall t\ \neg(i,t) \in \emptyset$$

$$(\textit{scons})\ \forall i,j,t,t',\Sigma\ ((i,t) \in \textit{scons}(j,t',\Sigma)\ \leftrightarrow\ (i = j \wedge t = t') \vee (i,t) \in \Sigma)$$

For the sake of readability, we abuse notation by using common set-theoretic notation in TDL formulas, writing for example $\Sigma \cup \{(i,t)\}$ instead of *scons*(i,t,Σ), and $\{(i,t),(j,t')\}$ instead of *scons*$(j,t',$*scons*$(i,t,\emptyset))$.

5.2 Motivating TDL

The formula $r^t_i\alpha$ intuitively means that at time t, i has access right of access type α (the object o which may be accessed with access type α is not made explicit in TDL).

As [4], we make a distinction between *belief* and *strong belief*: A principal who *believes* φ at time t (denoted $B^t_{i,\Sigma}\varphi$) has some justification for φ but believes that the justification might be wrong. A principal i who *strongly believes* φ at time t (denoted $K^t_{i,\Sigma}\varphi$) on the other hand believes that his/her/its justification for φ is correct. The Σ in the subscript of the belief operators indicates whether the belief is explicit or implicit (see Section 4.2 for this distinction): If Σ is $\emptyset$, it is explicit belief. If Σ is a non-empty set, the belief is implicit, and Σ indicates the principals who mediate this implicit belief together with the time points of their beliefs that mediate this belief. For example, if i trusts j at time t, j trusts k at time t', and k believes φ at time t_2, then i implicitly believes φ at time t, and this implicit belief is mediated by j and k through their beliefs at time t' and t_2 respectively: $B^t_{i,\{(j,t'),(k,t_2)\}}\varphi$.

Similarly as in [1], the fact that a principal i trusts a principal j on access α is formalized in TDL by a formula of the form $K^t_{i,\Sigma}T_j\alpha$. Here $T_j\alpha$ can be read intuitively as "j is trustworthy on access α". This way of formalizing the trust relation between two agents has the advantage of formally clarifying the difference between not trusting someone and actively distrusting someone, the first being formalized by $\neg K^t_{i,\Sigma}T_j\alpha$ (i.e. i lacks a strong belief about the trustworthiness of j), and the second by $K^t_{i,\Sigma}\neg T_j\alpha$ (i.e. i believes that j not trustworthy). Furthermore, weak distrust ($B^t_{i,\Sigma}\neg T_j\alpha$) is a useful formalization for the reserved kind of distrust that we have in Scenario 3.

TDL allows us to model five different levels of trust between a principal i and a principal j: *Strong trust*, where i strongly believes that j is trustworthy ($K^t_{i,\Sigma}T_j\alpha$), *weak trust* ($B^t_{i,\Sigma}T_j\alpha \wedge \neg K^t_{i,\Sigma}T_j\alpha$), *lack of trust* ($\neg B^t_{i,\Sigma}T_j\alpha \wedge \neg B^t_{i,\Sigma}\neg T_j\alpha$), *weak distrust* ($B^t_{i,\Sigma}\neg T_j\alpha \wedge \neg K^t_{i,\Sigma}\neg T_j\alpha$), and *strong distrust* ($K^t_{i,\Sigma}a\neg T_j\alpha$). The distinction between weak trust and lack of trust will not be relevant for modelling the reasoning about delegation and revocation, but the distinction between the remaining four levels of trust will be relevant.

Additionally to trust in someone on an action, the logic can also express epistemic trust: $K^t_{i,\Sigma}T^{t'}_j\varphi$ intuitively means that i trusts j not to make mistakes in judgements about the truth value of φ, if the judgement is made before time point t'. The time point at which the judgement was made needs to be considered in order to correctly model local revocations, in which an agent still trusts the authorizations previously produced by the revokee, but does not trust new authorizations issued by the revokee (see Scenario 2).

The action of granting to j the right to perform action α is modelled in TDL by the action of publicly announcing one's trust in j on action α. Whenever one makes a public announement, the announcement gets marked as an announcement of belief (B), strong belief (K), lack of belief ($\not{B}$) or lack of strong belief ($\not{K}$). [4] uses the letter I for the action of informing someone, which is similar to the action of public announcement; so we have decided to use the letter I to denote public announcements: For example, $I^t_iK\varphi$ intuitively means that i publicly announces its strong belief in φ at time t. $I^t_iKT_j\alpha$ means that i announces its strong trust in j on action α, and corresponds to i issuing a positive authorization for j with permission α at time t. Performing a Weak Global Delete revocation for permission α is achieved by the public announcement $I^t_i\not{B}T_j\alpha$, with which i retracts its trust in j by announcing that it no longer believes j to be trustworthy.

If i wanted to give j the right to give any principal k the right to perform access α, i could achieve this by publicly announcing its strong trust in j concerning judgements about the trustworthiness of other principals: $I^t_iK\ \forall k\ T^\infty_jT_k\alpha$. If i was trusted by the SOA to make such an announcement, then j would now be trusted by the SOA to announce its trust in any principal k on access α, i.e. to grant the right to perform access α to any principal. However, j would not yet be permitted to delegate to someone else the right to grant this right. To give j this right, the i's announcement would have to be $I^t_iK\ \forall k_1{}^\infty\ \forall k_2\ T^\infty_{k_1}T_{k_2}\alpha$. After this announcement, j can make an announcement of the form $I^{t'}_jK\ \forall k_2\ T^\infty_{k_1}T_{k_2}\alpha$, i.e. j can grant to some principal k_1 the right to grant to some further principal k_2 the right to perform access α.

This method can be used to model delegation with an arbitrary bound on the length for the delegation chain. But both Hagström et al.'s framework and our refinement of it do not put any bound on the length of delegation chains. In order to use this method for modelling delegation with no bound on the length of the delegation chain, we would have to allow principals to make infinitely many public announcements at once. In order to avoid this complication, we have introduced a third kind of trust, denoted $T_i\mathbb{D}\alpha$. Its intuitive meaning is that i is trusted to delegate the right to perform access α. Formally, its intended semantics is that $T_j\mathbb{D}\alpha$ should imply every formula in the infinite set $\{T_j\alpha, \forall k\ T^\infty_jT_k\alpha, \forall k_1T^\infty_j\ \forall k_2\ T^\infty_{k_1}T_{k_2}\alpha, \ldots\}$. This is achieved by the following axiom governing the behaviour of $T_i\mathbb{D}\alpha$:

$$(T\mathbb{D})\ T_i\mathbb{D}\alpha \rightarrow T_i\alpha \wedge T^\infty_iT_j\mathbb{D}\alpha$$

Using this new kind of trust, we can use the public announcement $I_iKT_j\mathbb{D}\alpha$ to model i's issuing a positive authorization for j with permission D at time t. $I^t_i\not{B}T_j\mathbb{D}\alpha$ models i's performing a Weak Global Delete revocation on j for permission $\mathbb{D}$.

Performing a P-t-p Global Resilient revocation can be modelled by announcing strong distrust in another principal: $I^t_iK\neg T_j\alpha$ or $I^t_iK\neg T_j\mathbb{D}\alpha$. If i explicitly announces its distrust j in this way, i thereby prevents an implicit belief in the trustworthiness of j to be passed through i to the SOA. Hence j will need to be connected to the SOA via some trust chain that is independent of i in order to get access or delegation right.

If i's strong distrust in j corresponds to a p-t-p revocation, what will correspond to a strong revocation? The answer is that for a strong revocation i needs to make an announcement that will ensure that the SOA does not trust j. For if the SOA is ensured not to trust j, then j's rights are blocked, just as after a strong revocation. Of course, in blocking j's rights in this way, i will have to make use of the fact that the SOA has – either directly or indirectly – granted i the right to perform a strong revocation. As a first attempt at modelling strong revocation and the right to perform strong revocations in TDL, it therefore makes sense to consider the following approach (for simplicity, we restrict ourselves to strong revocations for permission α):

APPROACH 1. *$I_i^t K \neg T_j \alpha$ models not only p-t-p revocation, but also strong revocation. The stronger effect of strong revocation is achieved by having the SOA believe in i's judgements of other principals non-trustworthiness when i has the strong revocation right. So i's issuing a positive authorization to j for permission S should be modelled by i announcing j to be trusted on distrusting other principals: $I_i^t K \forall k\ T_j^\infty \neg T_k \alpha$.*

The problem with this approach is that it would lead to blocked access in some situations where access should be granted. Suppose for example that the SOA grants A strong revocation right, A grants this right further to B, and B uses this right to issue a strong negative authorization to C. Furthermore, the SOA grants simple access right to A, who grants this further directly to C. So far, C does not have access, since its access is blocked by the strong negative authorization issued by B. But suppose next that the SOA globally revokes A's strong revocation right. Then B also loses its strong revocation right, so that the negative authorization issued by B becomes inactive. Hence C should now have access. But with the above approach, the fact that A granted B the strong revocation right means that A trusts B on distrusting other principals. Since B still distrusts C, this would mean that A implicitly distrusts C, so that C cannot have access based on a trust chain going through A.

To solve this problem, we model i's performing a Strong Global Resilient revocation on j for permission α by i announcing that the SOA should strongly distrust j: $I_i^t K \forall t\ K^t_{\text{SOA},\emptyset} \neg T_j \alpha$. Note that nested belief modalities are interpreted in a deontic way: $K^t_{i,\emptyset} K^t_{j,\emptyset} \varphi$ means that i strongly believes that j *should* strongly believe that φ. Granting i strong delegation right should make i being trusted on judgements about the SOA's strong distrust in other principals. In order to also be able to model performing a Strong Global Resilient revocation for permission S, we need to introduce a fourth kind of trust denoted $T_i \mathbb{S} \alpha$, for reason similar as the reasons for introducing $T_i \mathbb{D} \alpha$. The intuitive meaning of $T_i \mathbb{S} \alpha$ is that i is trusted in judgements about the SOA's strong distrust in other principals; here we need to allow for public announcements of distrust of various kinds: $\neg T_j \alpha$, $\neg T_j \mathbb{D} \alpha$ and $\neg T_j \mathbb{S} \alpha$. Furthermore, $T_i \mathbb{S} \alpha$ should imply that i is trusted to delegate the right to perform strong revocations, i.e. to consider another principal k trustworthy for performing strong revocations. The following axiom captures all this:

$$(T\mathbb{S})\ T_i \mathbb{S} \alpha \rightarrow T_i^\infty T_j \mathbb{S} \alpha \wedge T_i^\infty \forall t\ K^t_{\text{SOA},\emptyset} \neg T_j \alpha \wedge T_i^\infty \forall t\ K^t_{\text{SOA},\emptyset} \forall k\ \forall t_2\ \neg T_j^{t_2} T_k \mathbb{D} \alpha \wedge T_i^\infty \forall t\ K^t_{\text{SOA},\emptyset} \neg T_j \mathbb{S} \alpha$$

A principal i may epistemically trust both a principal who believes φ and a principal who believes $\neg\varphi$. In such a situation we do not want i to implicitly hold the inconsistent beliefs that φ and that $\neg\varphi$, because we want implicit belief to stay consistent. Instead, we want i to implicitly believe neither φ nor $\neg\varphi$. So the principle that i's epistemic trust in j concerning φ and j's belief in φ together imply i's implicit belief in φ cannot hold without exception. Instead, we say that if i epistemically trusts j concerning φ and j believes φ, then i has a *reason* to believe φ. To deduce that i believes φ from the fact that i has a reason to believe φ we additionally require there to be no reason for i to believe $\neg\varphi$. In TDL, $R^t_{i,\Sigma} B\varphi$ (respectively $R^t_{i,\Sigma} K\varphi$) denotes the fact that at time t, i has a reason to believe (respectively to strongly believe) φ implicitly, mediated by Σ. In order for the absence of a reason for i to believe $\neg\varphi$ to be provable, it needs to be formulated using negation-as-failure rather than classical negation: $\sim\exists\Sigma\ R^t_{i,\Sigma} B \neg\varphi$.

5.3 TDL proof theory

In order to correctly capture the intended functioning of negation-as-failure in TDL's proof theory, we need TDL's deducibility relation $\Gamma \vdash \varphi$ to be defined in such a way that $\Gamma \nvdash \varphi$ in general implies $\Gamma \vdash \sim\varphi$. This can be achieved by defining this deducibility relation inductively[2] as follows:

DEFINITION 5. *We define $\Gamma \vdash \varphi$ to be the case if one of the following conditions holds:*

- $\varphi \in \Gamma$
- *φ is an axiom of TDL*
- *For some formula ψ, $\Gamma \vdash \psi$ and $\Gamma \vdash \psi \rightarrow \varphi$ (modus ponens)*
- *φ is of the form $K^t_{i,\Sigma}\psi$ and $\vdash \psi$ (necessitation for strong belief)*
- *φ is of the form $R^t_{i,\Sigma} K\psi$ and $\vdash \psi$ (necessitation for reasons for strong belief)*
- *φ is of the form $\sim\psi$, where ψ is not of the form $\sim\chi$, and $\Gamma \nvdash \psi$ (negation-as-failure)*

The axioms of TDL include the axioms of the standard Hilbert system for first-order logic (as described for example in subchapter 3.6 of [10]) as well as all axioms mentioned in section 5.2 and in the rest of this section.

The axioms governing the behaviour of the two belief modalities and their interaction are taken over from Demelombe [4]. Both belief modalities obey the system (KD):

$$(\mathrm{K}_B)\ B^t_{i,\Sigma}\varphi \wedge B^t_{i,\Sigma}(\varphi \rightarrow \psi) \rightarrow B^t_{i,\Sigma}\psi$$
$$(\mathrm{D}_B)\ \neg(B^t_{i,\Sigma}\varphi \wedge B^t_{i,\Sigma}\neg\varphi)$$
$$(\mathrm{K}_K)\ K^t_{i,\Sigma}\varphi \wedge K^t_{i,\Sigma}(\varphi \rightarrow \psi) \rightarrow K^t_{i,\Sigma}\psi$$
$$(\mathrm{D}_K)\ \neg(K^t_{i,\Sigma}\varphi \wedge K^t_{i,\Sigma}\neg\varphi)$$

Furthermore, strong belief satisfies the axiom schema (KT), which intuitively says that a principal strongly believes that what it strongly believes is true:

$$(\mathrm{KT})\ K^t_{i,\Sigma}(K^t_{i,\emptyset}\varphi \rightarrow \varphi)$$

Strong belief implies weak belief:

$$(KB)\ K^t_{i,\Sigma}\varphi \rightarrow B^t_{i,\Sigma}\varphi$$

We need axioms similar to these axioms about the two belief modalities for the reason-for-belief modality:

$$(\mathrm{K}_{RB})\ R^t_{i,\Sigma} B\varphi \wedge R^t_{i,\Sigma} B(\varphi \rightarrow \psi) \rightarrow R^t_{i,\Sigma} B\psi$$
$$(\mathrm{K}_{RK})\ R^t_{i,\Sigma} K\varphi \wedge R^t_{i,\Sigma} K(\varphi \rightarrow \psi) \rightarrow R^t_{i,\Sigma} K\psi$$
$$(\mathrm{KT}_R)\ R^t_{i,\Sigma} K K^t_{i,\emptyset} K\varphi \rightarrow R^t_{i,\Sigma} K\varphi$$
$$(RKRB)\ R^t_{i,\Sigma} K\varphi \rightarrow R^t_{i,\Sigma} B\varphi$$

[2]Since Definition 5 is an inductive definition, it can – similarly to Definitions 2 and 3 discussed in Appendix A – in some contexts lead to the relation $\vdash$ being undefined. However, as follows from the proof of Theorem 1 sketched in Appendix C, $\Gamma \vdash \varphi$ is defined whenever Γ is a set of announcement formulas corresponding to a authorization specification free of strong S-revocation loops.

Recall that $B^t_{i,\Sigma}B^{t'}_{j,\Sigma'}\varphi$ is interpreted to mean that i believes that j *should* believe that φ. It is reasonable to assume that i believes that someone else should believe φ iff i herself believes φ. This is captured by the following four axiom schemas:

$$(BB_1)\ B^t_{i,\Sigma}\varphi \rightarrow B^t_{i,\Sigma}\forall t' B^{t'}_{j,\emptyset}\varphi$$
$$(BB_2)\ B^t_{i,\Sigma}B^{t'}_{j,\emptyset}\varphi \rightarrow B^t_{i,\Sigma}\varphi$$
$$(KK_1)\ K^t_{i,\Sigma}\varphi \rightarrow K^t_{i,\Sigma}\forall t' K^{t'}_{j,\emptyset}\varphi$$
$$(KK_2)\ K^t_{i,\Sigma}K^{t'}_{j,\emptyset}\varphi \rightarrow K^t_{i,\Sigma}\varphi$$

The action of asserting a strong belief is always also considered an action of asserting the corresponding weak belief, and the action of denying a weak belief is always also considered an action of denying the corresponding strong belief:

$$(IKB)\ I^t_i K\varphi \rightarrow I^t_i B\varphi$$
$$(IBK)\ I^t_i \not B\varphi \rightarrow I^t_i \not K\varphi$$

Since the SOA has the ultimate authority over the object in question, every principal has the right to perform an action iff it is trusted by the SOA on that action:

$$\text{(SOA)}\ r^t_i a \leftrightarrow \exists\Sigma\ K^t_{\text{SOA},\Sigma}T_i\alpha$$

If at time t, i has epistemic trust in j concerning the judgements about φ made before time point t', this means that if j believes φ at time $min(t,t')$, i generally has a reason to believe φ. However, this reason to believe φ cannot be inferred if j believes φ only implicitly, mediated by the belief of some principal k at time t, and i has a reason to distrust k concerning judgements about φ held at time t. This is formalized in the *axiom schemas of epistemic trust*:

$$\begin{aligned}(\text{ET}_B)\ & B^t_{i,\emptyset}T^{t'}_j\varphi \wedge ((t' > t \wedge t_2 = t) \vee (\neg t' > t \wedge t_2 = t')) \wedge \\ & B^{t_2}_{j,\Sigma}\varphi \wedge \sim\exists k, t_3, \Sigma'\ (k \in \Sigma \wedge R^t_{i,\Sigma'}B\neg T^{t_3}_k\varphi) \\ & \rightarrow\ R^t_{i,\Sigma\cup\{(j,t_2)\}}B\varphi\end{aligned}$$

$$\begin{aligned}(\text{ET}_K)\ & K^t_{i,\emptyset}T^{t'}_j\varphi \wedge ((t' > t \wedge t_2 = t) \vee (\neg t' > t \wedge t_2 = t')) \wedge \\ & K^{t_2}_{j,\Sigma}\varphi \wedge \sim\exists k, t_3\Sigma'\ (k \in \Sigma \wedge R^t_{i,\Sigma'}B\neg T^{t_3}_k\varphi) \\ & \rightarrow\ R^t_{i,\Sigma\cup\{(j,t_2)\}}K\varphi\end{aligned}$$

The following axioms govern the relationship between belief and reason to belief that we already explained at the end of Section 5.2:

$$(RB)\ R^t_{i,\Sigma}B\varphi \wedge \sim\exists\Sigma'\ R^t_{i,\Sigma'}B\neg\varphi \rightarrow B^t_{i,\Sigma}\varphi$$
$$(RK)\ R^t_{i,\Sigma}K\varphi \wedge \sim\exists\Sigma'\ R^t_{i,\Sigma'}B\neg\varphi \rightarrow K^t_{i,\Sigma}\varphi$$
$$(BR)\ B^t_{i,\Sigma}\varphi \rightarrow R^t_{i,\Sigma}B\varphi$$
$$(BR)\ K^t_{i,\Sigma}\varphi \rightarrow R^t_{i,\Sigma}K\varphi$$

We assume that principals are sincere, in the sense that in general a principal believes what he/she/it has previously announced. However, this principle needs some restrictions: Firstly, a principal can distance itself from a previous annoncement by making an announcement with the same content as before but with opposite announcement modality (e.g., to distance itself from its previous announcement of belief in φ, a principal can announce its non-belief in φ). Secondly, an announcement of weak belief in φ can be made obsolete by an announcement of strong belief in $\bar{\varphi}$ by a trustworthy principal. Thirdly, an announcement of strong belief only implies that the principal has reasons for strong belief; strong belief can be implied using axiom (RK) in the absence of reasons for the negation. This is formalized in the *sincerity axiom schemas*:

$$\begin{aligned}(\text{Sin}_B)\ & I^{t'}_i B\varphi \wedge t > t' \wedge \sim\exists t_2 > t'\ (t > t_2 \wedge I^{t_2}_i \not B\varphi) \wedge \\ & \sim\exists j, \Sigma\ \exists t_3 > t\ (t > t_3 \wedge B^{t_3}_{i,\Sigma}T^t_j\bar{\varphi} \wedge I^{t_3}_j K\bar{\varphi}) \rightarrow B^t_{i,\emptyset}\varphi\end{aligned}$$

$$(\text{Sin}_K)\ I^{t'}_i K\varphi \wedge t > t' \wedge\ \sim\exists t_2 > t'\ (t > t_2 \wedge I^{t_2}_i \not K\varphi) \rightarrow R^t_{i,\emptyset}K\varphi$$

$$(\text{Sin}_{\not B})\ I^{t'}_i \not B\varphi \wedge t > t' \wedge\ \sim\exists t_2 > t'\ (t > t_2 \wedge I^{t_2}_i B\varphi) \rightarrow \neg B^t_{i,\emptyset}\varphi$$

$$(\text{Sin}_{\not K})\ I^{t'}_i \not K\varphi \wedge t > t' \wedge\ \sim\exists t_2 > t'\ (t > t_2 \wedge I^{t_2}_i K\varphi) \rightarrow \neg K^t_{i,\emptyset}\varphi$$

We assume that all principals trust themselves, as stated by the *axiom of self-trust*:

$$\text{(ST)}\ K^t_{i,\emptyset}T_i\alpha$$

Since $T^t_k\varphi$ means that k's judgements made before time point t about φ are trusted, $T^t_k\varphi$ should clearly imply $T^{t'}_k\varphi$ if t' is an earlier time point than t:

$$(T^t)\ T^t_k\varphi \wedge t > t' \rightarrow T^{t'}_k\varphi$$

In order for the binary relation $>$ to function properly in the logic, we need the following axiom scheme. For any two time point constants $c_1, c_2 \in \mathbb{Z} \cup \{\infty\}$, if $c_1 < c_2$ in the natural ordering of $\mathbb{Z} \cup \{\infty\}$ (in which ∞ is larger than any integer), the following formula is a TDL axiom:

$$(>)\ c_1 < c_2 \wedge \neg c_2 < c_1 \wedge \neg c_1 < c_1$$

6. SCENARIOS IN TDL

In this section we show how TDL can be used to model the reasoning about trust and distrust involved in justifying the choices of revocation schemes in the scenarios from section 4.1.

In order to formalize scenario 3, we need to add some details to the description of the scenario: Suppose that A is the SOA and that at time point 1, A grants C delegation right concerning the access α, i.e. $I^1_{\text{A}}KT_{\text{C}}\mathbb{D}\alpha$. At time 2, C grants B this delegation right: $I^2_{\text{C}}KT_{\text{B}}\mathbb{D}\alpha$. Later, let's say at time point 9, A finds out that C is leaving to join the rival company, and hence now distrusts C concerning access α or to grant delegation right concerning access a to anyone else: $I^9_{\text{A}}K\neg T_{\text{C}}\alpha$ and $I^9_{\text{A}}K\neg T_{\text{C}}\mathbb{D}\alpha$. A also explicitly denies her previous trust statement to make clear it is no longer in place: $I^9_{\text{A}}\not KT_{\text{C}}\mathbb{D}\alpha$. But since C never misused any rights, A still trusts the delegation authorizations issued by C before time point 9: $I^9_{\text{A}}K\ \forall k\ T^9_{\text{C}}T_k\mathbb{D}\alpha$. We expect that C loses his access and delegation rights at time point 9, but that B retains these rights.

We now explain how this expected result is actually attained in TDL: By axiom (Sin_K), $I^1_{\text{A}}KT_{\text{C}}\mathbb{D}\alpha$ and the fact that A does not deny this announcement before time point 9 imply that for $1 \leq t \leq 8$, $K^t_{\text{A},\emptyset}T_{\text{C}}\mathbb{D}\alpha$, which by $(T\mathbb{D})$ and (K_K) further implies $K^t_{\text{A},\emptyset}T_{\text{C}}a$. Since $A = \text{SOA}$, axiom (SOA) implies $r^t_C a$ and $r^t_C\mathbb{D}\alpha$, i.e. C has access and delegation rights from time point 2 until time point 8. But since $I^9_{\text{A}}\not KT_{\text{C}}\mathbb{D}\alpha$, we cannot deduce $r^9_C a$ and $r^9_C\mathbb{D}\alpha$ in this way: At time point 9, C no longer has access and delegation right, as expected. However, for $2 \leq t \leq 9$, we can deduce $r^t_B a$ and $r^t_B\mathbb{D}\alpha$, i.e. that B has access and delegation rights: First, note that for $1 \leq t \leq 9$ we have $K^t_{\text{A},\emptyset}\ \forall k\ T^9_{\text{C}}T_k\mathbb{D}\alpha$ (which by (K_K) implies $K^t_{\text{A},\emptyset}T^9_{\text{C}}T_{\text{B}}\mathbb{D}\alpha$). In case $1 \leq t \leq 8$, this follows from $K^t_{\text{A},\emptyset}T_{\text{C}}\mathbb{D}\alpha$, $(T\mathbb{D})$ and (K_K); in case $t = 9$, it follows from $I^9_{\text{A}}K\ \forall k\ T^9_{\text{C}}T_k\mathbb{D}\alpha$ and (Sin_K). $I^2_{\text{C}}KT_{\text{B}}\mathbb{D}\alpha$ and (Sin_K) imply that for $2 \leq t \leq 9$, $K^t_{\text{C}}T_{\text{B}}\mathbb{D}\alpha$, so using (ET_K), we can derive $K^t_{\text{A},\emptyset}T_{\text{B}}\mathbb{D}\alpha$, which similarly as in the above proofs of $r^t_C a$ and $r^t_C\mathbb{D}\alpha$ implies $r^t_B a$ and $r^t_B\mathbb{D}\alpha$.

Here is how the other scenarios discussed in Section 4 can be formalized in TDL (we assume that the revocation always takes place at time point 9):

Scenario 1. User A distrusts user C concerning access and delegation: $I^9_{\mathrm{A}} K \neg T_{\mathrm{C}} \mathbb{D}\alpha$. This implies not only $K^9_{\mathrm{A},\emptyset} \neg T_{\mathrm{C}} \mathbb{D}\alpha$, but by axiom (KK_1) also $K^9_{\mathrm{A},\emptyset} \forall t\ K^t_{\mathrm{SOA},\emptyset} \neg T_{\mathrm{C}} \mathbb{D}\alpha$. According to the explanations about strong revocations in section 5.2, if the SOA trusts A on strong revocations ($\exists \Sigma\ K^9_{\mathrm{SOA},\Sigma} T_{\mathrm{A}} \mathbb{S}\alpha$), the latter formula has the same effect as a Strong Global Resilient revocation.

Scenario 3. The reserved kind of distrust resulting from hearing a rumour for which one does not know whether it is true is modelled in TDL as weak distrust, i.e. weak belief in the non-trustworthiness of the principal in question: $I^9_{\mathrm{A}} B \neg T_{\mathrm{B}} \mathbb{D}\alpha$. Since axiom (Sin_B) blocks the inference of $B^t_{i,\emptyset} B \neg T_{\mathrm{B}} \mathbb{D}\alpha$ from $I^9_{\mathrm{A}} B \neg T_{\mathrm{B}} \mathbb{D}\alpha$ if some trusted principal announces trust in j (i.e. delegates to j), $I^9_{\mathrm{A}} B \neg T_{\mathrm{B}} \mathbb{D}\alpha$ loses its effect as soon as such an anouncement takes place. Hence we have the effect of a non-resilient revocation, as desired.

Scenario 4. User A neither trusts nor distrusts user C: $I^9_{\mathrm{A}} \not B T_{\mathrm{C}} \mathbb{D}\alpha$ and $I_{\mathrm{A}} \not B^9 \neg T_{\mathrm{C}} \mathbb{D}\alpha$. These announcements remove the effect of any previous announcement made by A about the trustworthiness of C, i.e. the situation is now practically the same as if A had never trusted C. This corresponds to a Weak Global Delete, in which a positive authorization is removed, leaving no trace of it ever having been there.

7. DESIRABLE BEHAVIOUR OF REVOCATION SCHEMES

In this section we show how TDL can be used to formally formulate a desirable property for a graph-theoretically defined revocation framework. This allows us to study revocation frameworks using the axiomatic method originating in social choice theory.

There are different revocation schemes because there are different reasons for revocating. We start this section by exhibiting a correspondence between revocation schemes and reasons for revocating formalizable in TDL. The main idea behind the desirable property of revocation frameworks that we define is that if performing revocation schemes and granting rights was replaced by publicly announcing one's formal reasons for revocating or granting, then these public announcements should logically imply (in TDL) a principal's access right iff that principal is actually granted access based on the delegation graph.

7.1 Matching reasons for revocating to revocation schemes

As explained in Section 5, there are five levels of trust that an agent can have in another agent, of which four need to be distinguished in modelling delegation and revocation. But even when i explicitly strongly distrusts j concerning delegation right ($K^t_{i,\emptyset} \neg T_j \mathbb{D}\alpha$), i may still trust j's previous judgements concerning $T_k \mathbb{D}\alpha$ for other principals k ($K^t_{i,\emptyset} \forall k T^t_j T_k \mathbb{D}\alpha$). So the level of trust in another agent concerning delegation right can be different from the level of trust concerning previously granted authorizations. However, these two levels of trust are not completely independent of each other: For example, $K^t_{i,\emptyset} T_j \mathbb{D}\alpha$ implies $K^t_{i,\emptyset} \forall k T^t_j T_k \mathbb{D}\alpha$. More generally, the second level of trust must be at least as high as the first level of trust. This means that only 10 of the 16 ensuing combinations of trust levels are actually possible.

Table 1 shows which granting-revocation behaviour corresponds to each of these ten possible combinations of trust levels. Some cells contain multiple revocation schemes. This means that the granting-revocation behaviour corresponding to the combination of trust levels represented by that cell consists of performing multiple revocation schemes at the same time. For an agent without strong revocation rights, the granting-revocation behaviour corresponding to some combination of trust levels is determined by dropping the strong revocations from the revocations in the cell that represent that combination of trust levels.

The formulas in the table have $\bar{\pi}$ in place of α, $\mathbb{D}\alpha$ or $\mathbb{S}\alpha$. $\bar{\pi}$ is defined as follows:

DEFINITION 6. *For $\pi \in \{A, D, S\}$, we define $\bar{\pi}$ by setting $\bar{\pi} := \alpha$ if $\pi = A$, $\bar{\pi} := \mathbb{D}\alpha$ if $\pi = D$, and $\bar{\pi} := \mathbb{S}\alpha$ if $\pi = S$.*

The revocation schemes in the table should always be for the same π that is used in the $\bar{\pi}$ in the formulas.

We consider the pair of levels of trust that i has in j to be the reason i has for granting a right to j or revoking a right from j. Hence the graph-theoretic definitions of the revocation schemes should be such that access is granted whenever this is justifiable on the basis of these trust-based reasons for granting and revocating. We use deducibility in TDL as our formal criterion for justifiablity.

These explanations already determine the desirable property for a set of graph-theoretic definitions of revocation schemes. We now proceed to formalizing this desirable property.

7.2 Formal desirable property

We first define a set $\mathbf{C}$ corresponding to the ten meaningful cells of Table 1:

DEFINITION 7. $\mathbf{C} := \{(m, n) \in \{1, 2, 3, 4\}^2 \mid m \geq n\}$.

Next we define a *granting-revocation action* corresponding to a cell in the table:

DEFINITION 8. *For $i, j \in \mathbf{S}$, $(m, n) \in \mathbf{C}$ and $\pi \in \mathbf{P}$, define $\mathbf{GR}(i, j, (m, n), \pi)$ to be the granting-revocation behavior in the cell in row m and column n of Table 1 performed by i onto j for permission π.*

For example, $\mathbf{GR}(\mathrm{A}, \mathrm{B}, (2, 2), D)$ is a Weak Global Delete revocation from A to B for permission D.

Next we define a *public announcement* corresponding to a cell in the table:

DEFINITION 9. *For $i, j \in \mathbf{S}$, $(m, n) \in \mathbf{C}$ and $\pi \in \mathbf{P}$, define $\mathbf{I}(i, j, (m, n), \pi)$ to be the set of public announcements by i in trust in j for permission π according to the level of trust of row m and column n of Table 1.*

For example, $\mathbf{I}(\mathrm{A}, \mathrm{B}, (4, 1), \alpha)$ is $\{I^t_{\mathrm{A}} K \forall k T^t_{\mathrm{B}} T_k \bar{\pi}, I^t_{\mathrm{A}} K \neg T_{\mathrm{B}} \bar{\pi}\}$.

We now need to define the notion of an authorization specification resulting from a sequence of granting-revocation-actions.

DEFINITION 10. *Given a sequence σ of elements of $\mathbf{S} \times \mathbf{S} \times \mathbf{C} \times \mathbf{P}$, we define the authorization specification $\mathbf{A}(\sigma)$ inductively as follows:*

- $\mathbf{A}(\langle\rangle) = \emptyset$
- *$\mathbf{A}(\langle (i_1, j_1, a_1, r_1), \ldots, (i_n, j_n, a_n, r_n)\rangle)$ is the authorization specification resulting from performing $\mathbf{GR}(i_n, j_n, a_n, r_n)$ on $\mathbf{A}(\langle (i_1, j_1, a_1, r_1), \ldots, (i_{n-1}, j_{n-1}, a_{n-1}, r_{n-1})\rangle)$.*

Now we need to define which sequences of granting-revocation-actions are actually valid in our system:

DEFINITION 11. *A sequence $\langle (i_1, j_1, a_1, r_1), \ldots, (i_n, j_n, a_n, r_n)\rangle$ of elements of $\mathbf{S} \times \mathbf{S} \times \mathbf{C} \times \mathbf{P}$ is called a* valid granting-revocation pattern *iff for every $k < n$, the authorization specification $\mathbf{A}(\langle (i_1, j_1, a_1, r_1), \ldots, (i_k, j_k, a_k, r_k)\rangle)$ is free of strong S-revocation loops and authorizes i to perform $\mathbf{GR}(i_k, j_k, a_k, r_k)$.*

	$K^t_{i,\emptyset}\forall k T^t_j T_k \bar{\pi}$	$\neg B^t_{i,\emptyset}\forall k T^t_j T_k \bar{\pi} \wedge$ $\neg B^t_{i,\emptyset}\forall t' \forall k \neg T^{t'}_j T_k \bar{\pi}$	$B^t_{i,\emptyset}\forall t' \forall k \neg T^{t'}_j T_k \bar{\pi} \wedge$ $\neg K^t_{i,\emptyset}\forall t' \forall k \neg T^{t'}_j T_k \bar{\pi}$	$K^t_{i,\emptyset}\forall t' \forall k \neg T^{t'}_j T_k \bar{\pi}$
$K^t_{i,\emptyset} T_j \bar{\pi}$	grant permission π	X	X	X
$\neg B^t_{i,\emptyset} T_j \bar{\pi} \wedge \neg B^t_{i,\emptyset} \neg T_j \bar{\pi}$	WLD	WGD	X	X
$B^t_{i,\emptyset} \neg T_j \bar{\pi} \wedge \neg K^t_{i,\emptyset} \neg T_j \bar{\pi}$	PLN $\circ$ SLN	WGD $\circ$ PLN $\circ$ SLN	PGN $\circ$ SGN	X
$K^t_{i,\emptyset} \neg T_j \bar{\pi}$	PLR $\circ$ SLR	WGD $\circ$ PLR $\circ$ SLR	PGN $\circ$ PLR $\circ$ SGN $\circ$ SLR	PGR $\circ$ SGR

Table 1: The correspondence between the revocation framework and reasons for revocating formalized in TDL

The notion of being free of strong S-revocation loops is explained and formally defined in Appendix A.

The following theorem, whose proof is sketched in Appendix C, now formally expresses that our refined revocation framework has the desirable property that we have previously already explained and motivated:

THEOREM 1. *Let $n \in \mathbb{N}$, and let σ be a valid granting-revocation pattern of length n. Then for all $i \in \mathbf{S}$, $\mathbf{I}(\sigma) \models r^n_i \alpha$ iff i is the SOA or there is an active authorization of the form $(p, i, +, \alpha, t)$ in $\mathbf{A}(\sigma)$.*

Note that if we had refrained from implementing in our revised framework one of the five changes to Hagström et al.'s framework discussed in section 3.1, the resulting framework would not satisfy this desirably property.

8. FUTURE WORK

In this paper we studied revocation for a version of delegation that does not have any bound on the length of delegation chains. However, TDL lends itself very well also to delegation with such a bound. Indeed, for this purpose a somewhat reduced version of TDL, which lacks the $T_i\mathbb{D}$ and $T_i\mathbb{S}$ trust operators, would be sufficient. It would be interesting to study the possibility to define a systematic revocation framework for such bounded delegation that satisfies a desirable property analogous to the one that our framework for revoking unbounded delegation has been shown to satisfy.

Some reasons for revoking rights cannot be captured in TDL: User A may revoke B's rights just because she does not consider it to be useful for B to have the right in question, even though she strongly trusts B. And A may choose a certain kind of revocation scheme based on considerations of responsibility, not just considerations of trust. Our preliminary investigations into this topic suggest that our refined framework also corresponds well to these not trust-based reasons for revocating, but further investigations are due in order to develop an extension of TDL that can formalize such reasons and proof our system to satisfy a desirable property based on this extended logic.

In order to develop an axiomatic theory of revocation schemes similar to the application of the axiomatic method in social choice theory, other desirable properties of revocation schems or revocation frameworks need to be identified an compared to the desirable property that we proposed.

9. CONCLUSION

After identifying some problems with Hagström et al.'s [8] revocation framework, we presented a refined framework that avoids these problems. In order to ensure that our refined framework does not itself suffer from similar problems, we systematically studied the relation between the reasons for revocating and the graph-theoretic definitions of revocation schemes. In order to formalize reasons for revocating based on trust and distrust, we developed Trust Delegation Logic (TDL). TDL allowed us to formulate a desirable property that a graph-theoretically defined revocation framework should satisfy. This desirable property is based on a correspondence between revocation schemes and reasons for revocating, and requires the revocation schemes to be defined in such a way that access is granted whenever this is justifiable on the basis of the reasons for granting and revocating. The main theorem of the paper asserts that our refined framework does satisfy this desirable property.

10. ACKNOWLEDGEMENTS

This work was supported by the FNR-FWO project *Specification logics and Inference tools for verification and Enforcement of Policies*.

11. REFERENCES

[1] G. Aucher, S. Barker, G. Boella, V. Genovese, and L. van der Torre. Dynamics in Delegation and Revocation Schemes: A Logical Approach. In Y. Li, editor, *Data and Applications Security and Privacy XXV*, volume 6818 of *Lecture Notes in Computer Science*, pages 90–105. Springer Berlin, 2011.

[2] E. Bertino, S. Jajodia, and P. Samarati. A Non-timestamped Authorization Model for Data Management Systems. In *Proceedings of the 3rd ACM Conference on Computer and Communications Security*, CCS '96, pages 169–178, New York, NY, USA, 1996. ACM.

[3] E. Bertino, P. Samarati, and S. Jajodia. An extended authorization model for relational databases. *Knowledge and Data Engineering, IEEE Transactions on*, 9(1):85–101, Jan 1997.

[4] R. Demolombe. Reasonig about trust: A formal logical framework. In C. Jensen, S. Poslad, and T. Dimitrakos, editors, *Trust Management*, volume 2995 of *Lecture Notes in Computer Science*, pages 291–303. 2004.

[5] M. Denecker. The Well-Founded Semantics Is the Principle of Inductive Definition. In J. Dix, L. del Cerro, and U. Furbach, editors, *Logics in Artificial Intelligence*, volume 1489 of *Lecture Notes in Computer Science*, pages 1–16. Springer Berlin Heidelberg, 1998.

[6] R. Fagin. On an Authorization Mechanism. *ACM Trans. Database Syst.*, 3(3):310–319, Sept. 1978.

[7] P. P. Griffiths and B. W. Wade. An Authorization Mechanism for a Relational Database System. *ACM Trans. Database Syst.*, 1(3):242–255, Sept. 1976.

[8] Å. Hagström, S. Jajodia, F. Parisi-Presicce, and D. Wijesekera. Revocations-A Classification. In *Proceedings of the 14th IEEE Workshop on Computer Security Foundations*, CSFW '01, pages 44–, Washington, DC, USA, 2001. IEEE Computer Society.

[9] B. Jayaraman and D. Jana. Set constructors, finite sets, and logical semantics. *The Journal of Logic Programming*, pages 55–77, 1999.

[10] W. Rautenberg. *A Concise Introduction to Mathematical Logic*. Springer, 2006.

[11] C. Ruan and V. Varadharajan. Resolving Conflicts in Authorization Delegations. In L. M. Batten and J. Seberry, editors, *ACISP*, volume 2384 of *Lecture Notes in Computer Science*, pages 271–285. Springer, 2002.

APPENDIX

A. INDUCTIVE INTERDEPENDENCE OF DEFINITIONS 2 AND 3

Definitions 2 and 3, in which we define the notions of an authorization being *active* and being *directly inactivated*, depend one on another. This interdependence looks similar to a problematic circular definition, but can actually be understood as an inductive definition, which can be formally interpreted using the well-founded semantics [5]. In this appendix we briefly sketch the theory of inductive definitions under the well-founded semantics, and comment about what practical consequences follow from using an inductive definition for defining wthe notion of an *active* authorization.

An *inductive definition* of predicates $P_1, \ldots, P_n$ consists of rules of the form $\forall \bar{x}\ P(\bar{x}) \leftarrow \psi$, where $P \in \{P^1, \ldots, P^n\}$ and ψ may itself refer to predicates in $\{P^1, \ldots, P^n\}$. In such a rule, $P(\bar{x})$ is called the *head* of the rule, and ψ the *body* of the rule.

An inductive definition may have a form, which makes it impossible to interpret the defined predicates in a way that is consistent with the rules. For example, the inductive definition $\{P \leftarrow \neg P\}$, which defined P to be the case if $\neg P$ is the case is problematic: If we assume P is false, then $\neg P$ is true, so by the single rule of this definition, P should be true. So P cannot be false. However, since P is defined through this inductive definition, it can only be true if some rule in the inductive definition is true. But the only rule in this definition cannot make P true.

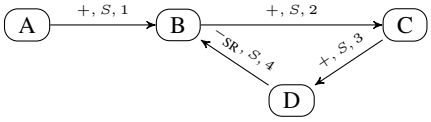
One approach to avoid such problems is to define some syntactic restrictions on the set of rules that ensure that such problems cannot arise. However, such syntactic restrictions are usually to restrictive, barring some inductive definitions that we can intuitively and formally make sense of.

Another approach to avoid such problems is that of using the *well-founded semantics* for inductive definitions, which is a partial semantics, i.e. for problematic definitions like the above, it does not define a truth function for the predicates that were purported to be defined by the inductive definition. But whenever an inductively defined predicate has an intuitively meaningful interpretation, the well-founded semantics formally assigns this interpretation to the predicate [5].

The well-founded semantics is defined through an inductive process which involves adding new information about the defined predicates at each step of the induction based on the rules in the inductive definition: This information consists of an assignment of truth values to *domain atoms*; a *domain atom* is a pair $(P, \bar{a})$ – usually written as $P(\bar{a})$ – where P is an n-ary predicate symbol defined in the inductive definition, and $\bar{a} \in D^n$, where D is the domain of the structure. At the first step of the induction, no truth values are assigned to domain conditions. At every subsequent step, the truth value **t** may be assigned to a domain atom $P(\bar{a})$ if applying the rules for P to the *previous* assignment of truth values to domain atoms establishes that $P(\bar{a})$ is true; the truth value **f** may be assigned to domain atoms $P_1(\bar{a_1})$, ..., $P_k(\bar{a_k})$ if applying the rules for $P_1, \ldots, P_k$ to the *resulting* assignment of truth values to domain atoms establishes that $P_1(\bar{a_1})$, ..., $P_k(\bar{a_k})$ are false. This different treatment of the two truth values reflects the inductive nature of the definition, according to which a domain atom can only be true if some rule makes it true, whereas a domain atom should be considered false in the absense of a rule making it true. The *well-founded model* of an inducive definition is defined to be the limit assignment of truth values to domain atoms in such an induction [5]. If the well-founded model does not assign either **t** or **f** to every domain atom of the defined predicates, it is a partial model.

The inductive definition of authorizations being *active* and *directly inactivated* can in some contexts lead to a partial well-founded model, i.e. to the notions of *active* and *directly inactivated* not having a coherently determinable meaning. Consider for example the following authorization specification:

EXAMPLE 8.



If we assume that the negative authorization from D to B is active, it directly inactivates the authorization from A to B. But then there is no active chain of authorizations that supports the authorization from D to B, so it would have to be inactive. If on the other hand we assume that the authorization from D to B is inactive, then the authorization from A to B is not directly inactivated, an the chain of authorization from A via B and C to D is active, thus ensuring that the authorization from D to to B is active. So either way we run into a contradiction. This means that the interpretation of "active" under the well-founded semantics is partial in the context of this authorization specification.

In this example, the problem was caused by principal D using its power to perform strong revocations in order to remove the rights from B, even though D's right to perform strong revocations depended on a delegation chain including B. The problem can be avoided by adding a constraint to the system that disallows principals from using their strong revocation right to remove the strong revocation right from a principal on whom they depend for their strong revocation right.

This constraint needs to be formulated in both a more formal and a more general way. For this we first need the notion of a $+-S$-chain, which formalizes the notion of a potentially active chain of positive S-authorizations followed by a strong negative S-authorization, which can only be inactivated if attacked by another $+-S$-chain. The time stamp of a $+-S$-chain indicates the least time stamp that a positive authorization must have in order not to be affected by the $+-S$-chain.

DEFINITION 12. *A $+-S$-chain with time stamp t is a chain of authorizations $(p_0, p_1, +, S, t_1)$, $(p_1, p_2, +, S, t_2)$, ..., $(p_{n-1}, p_n, +, S, t_n)$, $(p_n, p_{n+1}, \tau, S, t_{n+1})$ satisfying the following properties:*

- *$p_0 =$ SOA*
- *Either $\tau = -_{\text{SR}}$ and $t = \infty$, or $\tau = -_{\text{SN}}$ and $t = t_{n+2}$.*
- *There are no i, j with $0 \leq i < j \leq n$ such that there is an authorization $(p_i, p_j, -_{\text{PR}}, S, t')$.*
- *There are no i, j with $0 \leq i < j \leq n$ such that there is an authorization $(p_i, p_j, -_{\text{PN}}, S, t')$ with $t' < t_j$.*

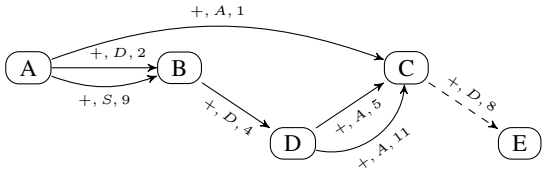
We say that a $+-S$-chain C_1 with time stamp t *attacks* a $+-S$-chain C_2 iff C_1 ends in a principal that has issued one of the authorizations in C_2 at some time $t' > t$.

Now the formalized and generalized constraint can be formulated as follows: We require that the $+-S$-chain in the authorization specification can be partially ordered in such a way that a $+-S$-chain C_1 attacks a $+-S$-chain C_2 only if $C_1 < C_2$ in

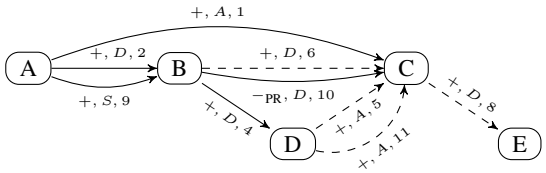
the partial ordering. Informally, this means that there should be no loops of $+-S$-chains under the attack relation. An authorization specification satisfying this constraint is called *free of strong S-revocation loops*.

B. FURTHER REVOCATION EXAMPLES

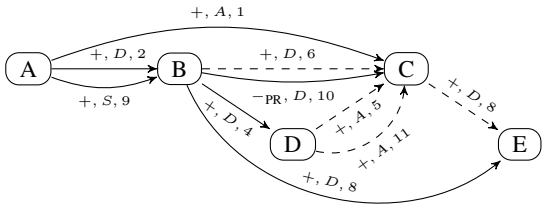
EXAMPLE 9. *Weak Global Delete revocation from* B *to* C



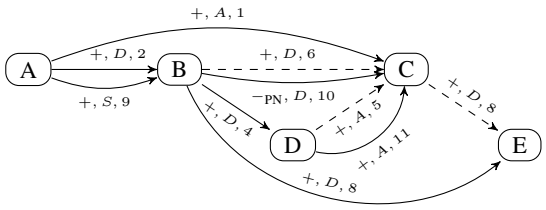
EXAMPLE 10. *P-t-p Global Resilient revocation from* B *to* C



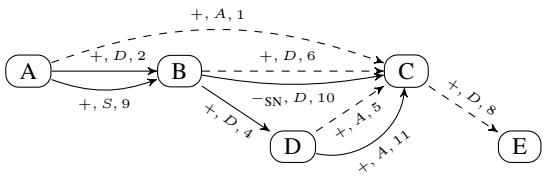
EXAMPLE 11. *P-t-p Local Resilient revocation from* B *to* C



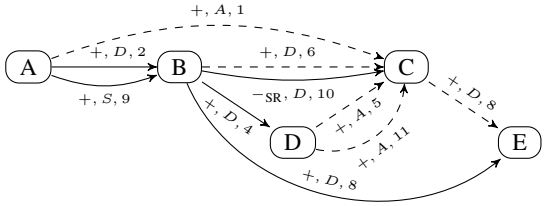
EXAMPLE 12. *P-t-p Local Non-resilient revocation from* B *to* C



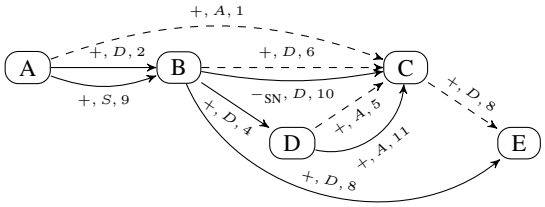
EXAMPLE 13. *Strong Global Non-resilient revocation from* B *to* C



EXAMPLE 14. *Strong Local Resilient revocation from* B *to* C



EXAMPLE 15. *Strong Local Non-resilient revocation from* B *to* C



C. SKETCH OF PROOF OF THEOREM 1

We prove Theorem 1 by first exhibiting a procedure for determining which authorizations in $\mathbf{A}(\sigma)$ are active, and then exhibiting a correspondence between statements about the activeness of authorization in $\mathbf{A}(\sigma)$ and statements about the deducibility of certain TDL formulas from $\mathbf{I}(\sigma)$.

First we need to define the notion of a π-chain for $\pi \in \mathbf{P}$:

DEFINITION 13. *A* π-chain *is a chain of authorizations* $(p_0, p_1, +, \pi', t_1)$, $(p_1, p_2, +, \pi', t_2)$, $\ldots$, $(p_{n-1}, p_n, +, \pi', t_n)$, $(p_n, p_{n+1}, +, \pi, t_{n+1})$ *satisfying the following properties:*

- $p_0 = SOA$
- $\pi' = D$ *if* $\pi = A$, *and* $\pi' = \pi$ *otherwise.*
- *There are no* i, j *with* $0 \leq i < j \leq n+1$ *such that there is an authorization* $(p_i, p_j, -_{\text{PR}}, \pi'', t')$, *where* $\pi'' = \pi$ *if* $j = n+1$, *and* $\pi'' = \pi'$ *otherwise.*
- *There are no* i, j *with* $0 \leq i < j \leq n+1$ *such that there is an authorization* $(p_i, p_j, -_{\text{PN}}, \pi'', t')$ *with* $t' < t_j$, *where* $\pi'' = \pi$ *if* $j = n+1$, *and* $\pi'' = \pi'$ *otherwise.*

Note that a $+-S$-chain is an S-chain followed by negative S-authorization.

σ is a valid granting-revocation pattern, so $\mathbf{A}(\sigma)$ is free of strong S-revocation loops, i.e. there is a partial ordering $<$ on the set of $+-S$-chains over $\mathbf{A}(\sigma)$. Since the set of $+-S$-chains over $\mathbf{A}(\sigma)$ is finite, $<$ is a well-ordering, and we can perform induction along $<$. This allows us to determine which $+-S$-chains are active: A $+-S$-chain C_1 is active iff it is not attacked by an active $+-S$-chain $C_2 < C_1$.

Next we can establish which positive S-authorizations are active: A positive S-authorization is directly inactivated iff its end node is attacked by an active $+-S$-chain. A positive S-authorization is active iff it is an element of an S-chain whose authorizations are not directly inactivated.

Next we can establish that a strong negative authorizations is active iff it starts at some principal at which some active S-chain ends. This allows us to establish which other authorizations are directly inactivated: $(i, j, +, \pi, t)$ is directly inactivated iff there is an active authorization (k, j, τ, π, t') such that either $\tau = -_{\text{SR}}$ or $\tau = -_{\text{SN}}$ and $t < t'$. Finally, we can establish that a positive π-authorization is active iff it is an element of a π-chain whose authorizations are not directly inactivated.

In a similar way as one can show in a step-by-step way the correctness of this procedure for determining the activeness of authorizations in $\mathbf{A}(\sigma)$, one can prove the following three equivalences:

1. For $\tau = -_{\text{SR}}$ or $\tau = -_{\text{SN}}$, (i, j, τ, π, t) is active at time t' iff $\mathbf{I}(\sigma) \vdash \exists\Sigma\ K^{t'}_{\text{SOA},\Sigma} T_i \mathbb{S}\alpha$.
2. $(i, j, +, \pi, t)$ is directly inactivated at time t' iff $\mathbf{I}(\sigma) \vdash \exists\Sigma\ R^{t'}_{\text{SOA},\Sigma} B \neg T_j \bar{\pi}$.
3. $(i, j, +, \pi, t)$ is active at time t' iff $\mathbf{I}(\sigma) \vdash \exists\Sigma\ K^{t'}_{\text{SOA},\Sigma} T_i \bar{\pi}$ and $\mathbf{I}(\sigma) \nvdash \exists\Sigma\ R^{t'}_{\text{SOA},\Sigma} B \neg T_j \bar{\pi}$.

The theorem now follows from equivalence 3 together with TDL axioms (SOA), (RK) and (ST).